WebAssembly vs native code

• 3 minutes to read

WebAssembly aims to drastically improve your application performance, safety, and developer productivity. It replaces native code with a managed container and finely grained security model.

Why do we program in native code in 2020?

In the past several years, CPU speed has pretty much stopped improving. At the same time, AI, big data, and blockchain have all create huge demands for more computing power. So far, the solution has been more and more native code in our software. Native code is efficient, close to the hardware, and can access specialized hardware such as GPU and AI chips.

However, native code also has issues such as platform dependency and safety. The big trend in software engineering in the past 30 years has been to move away from native code into managed code running inside virtual machines or containers.

How exactly is WebAssembly better than native code?

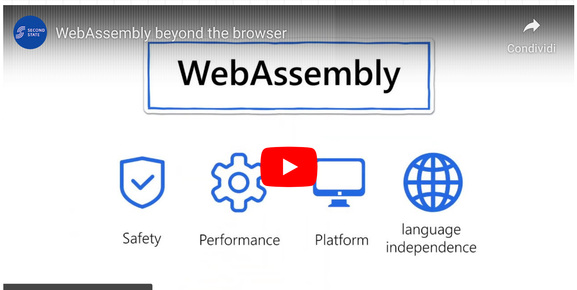
WebAssembly is the next-generation virtual machine that will help us turn native code modules into managed services. It's advantages include the following.

- Safety: WebAssembly bytecode is executed inside a virtual machine and cannnot crash the host system.
- Security: WebAssembly has a module security model that controls what the bytecode programs can access outside of the virtual machine.
- High performance: WebAssembly bytecode is highly efficient and runs very fast. It is on par with native code performance.
- Lightweight: WebAssembly virtual machines are very small (typically less than 10MB) and takes little memory to start and run.
- Portability: WebAssembly bytecode is cross-platform. It can run without change on all major operating systems and hardware platforms.
- Manageability: WebAssembly virtual machines can be started, stopped, and hot swapped using devops tools.
- Language agnostic: Multiple programming languages, not just C and C++, can compile into WebAssembly bytecode. In particular, Rust is well supported.
- Ecosystem support: WebAssembly programs can interoperate with popular programming langauages and frameworks, such as JavaScript, Python, and PHP.

WebAssembly is safe, fast, language-agnostic, and platform-independent.

Isn't WebAssembly mostly used inside the web browser?

WebAssembly started as a collaboration between Google, Mozilla, Apple, and Microsoft. It was envisioned to be a high-performance code execution engine inside browsers. The typical applications would be in-browser animated games that require performance, much like the Java Applet from the old days.



However, like Java and JavaScript before it, WebAssembly is finding success on the server-side. WebAssembly's safety, performance, platform, and language independence, make it an ideal server-side runtime.

Is it true that one must learn Rust in order to use WebAssembly?

No. WebAssembly is language agnostic. You can invoke WebAssembly programs and functions from a variety of different host languages, such as Javascript, Rust, Go, Python or even PHP.

You can also write WebAssembly programs in a variety of different programming languages. However, it is also true that Rust is currently the most widely used language to create WebAssembly programs and modules.

Rust has been voted the most beloved programming language for the past 4 years in a row. It is the hottest programming language right now. It has many exciting features. For example, it is powerful and flexible like C, but much safer and without Java's performance overhead. It supports both object-oriented and functional programming paradigms. It is one of the fastest-growing programming languages in the world and is now used in the entire software stack from front end to back-end to infrastructure.

WebAssembly #question-answer #serverless #function-as-a-service #ai-as-a-service